**DOSSIER DE PROJET**

**PETGAME**

SOMMAIRE

1. Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité
   1. Maquetter une application
   2. Réaliser une interface utilisateur web statique adaptable
   3. Développer une interface utilisateur web dynamique
   4. Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce
2. Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité
   1. Créer une base de données
   2. Développer la partie back-end d’une application web ou web mobile
   3. Développer les composants d’accès aux données
   4. Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

**Résumé de projet**

J'ai fait un projet qui regroupe tous les modules du CCP1et CCP2, il s'agit d'un site en deux parties qui porte sur le sujet des animaux domestiques.

J'ai tout d'abord établi un cahier des charges que j'ai modifié au cours du processus étant donné que des idées me venaient au fur et à mesure ; ensuite j'ai conçu un dictionnaire de données qui m'a permis de connaitre le nombre de tables dont j'aurais besoin pour le côté back-end de mon projet, puis je l'ai transformé en modèle conceptuel de données qui correspond au CP5. Par la suite j'ai pu créer ma base de données et mes tables en accord avec les cardinalités définies dans le MCD via du code SQL ce qui correspond au CP6.

Enfin j'ai pu faire ma maquette sur Photoshop qui montre les pages et la forme assez générale de mon projet ce qui correspond au CP1. La première partie s'inspire du site « Le Bon Coin » et à l'instar de celui propose des annonces commerciales entre particuliers ; la plus grosse différence est que mon site autorise uniquement des animaux domestiques sur les annonces pour permettre une transaction.

Les langages utilisés pour cette première partie sont HTML/CSS, Javascript, PHP qui correspondent au CP2, CP3 et CP7.

La seconde partie quant à elle est une extension de mon site qui a totalement été faite sur le CMS (Content Management System) Wordpress, en l'installant du coté back-end.

Cette partie comporte deux fonctionnalités qui sont en réalité des plugins dont un créé par moi-même. Le premier est le plugin wooCommerce qui va me permettre de vendre des produits alimentaires ou des jouets pour animaux domestiques et le second est la page contact qui permettra aux clients d'envoyer un mail directement à l’administrateur du site quel qu’en soit la raison.

**STAGE**

J'ai eu la chance de faire un stage de deux mois et demi qui porte sur un ensemble de quatre infrastructures nommée MERN (MongoDB, Express.js, React.js, Node.js). Ce sont des technologies qui appartiennent au langage front-end JavaScript, chacun d'entre eux va pouvoir interagir avec l'autre pour avoir comme finalité de créer un site Web dans sa globalité c'est à dire côté back-end (côté serveur) et front-end (côté client), chose qui n'est pourtant pas possible avec du langage JavaScript étant donné qu'il est uniquement côté client mais bien évidemment si on utilise quatre fonctionnalités c'est que certaines disposent de quoi faire tourner le site côté back-end donc me voilà maintenant dans l'explication des différents rôles de chacune de ces fonctionnalités plus ou moins détaillée.

* **MongoDB** est un système de gestion de base de donnée objet qui stocke les données sous format JSON en voici un exemple :

{

"id": 1,

"name": "A wooden door",

"price": 12.50,

"tags": "home"

}

* **Express.js** est un serveur framework uniquement utilisé sous Node.js qui va permettre de structurer les pages web côté serveur.

const app = express();

app.use(bodyParser.json());

app.use(

cookieSession({

maxAge: 30\*24\*60\*60\*1000,

keys: [keys.cookieKey]

})

);

app.use(passport.initialize());

app.use(passport.session());

require('./routes/authRoutes')(app);

require('./routes/BillingRoutes')(app);

require('./routes/surveyRoutes')(app);

if(process.env.NODE\_ENV === 'production'){

app.use(express.static('client/build'));

const path = require('path');

app.get('\*', (req, res) =>{

res.sendFile(path.resolve(\_\_dirname, 'client', 'build', 'index.html'));

});

}

const PORT = process.env.PORT || 5000;

app.listen(PORT);

* **React.js** est la seule fonctionnalité côté front-end parmi les quatre cités : c'est une bibliothèque JavaScript avec ses propres composants, méthodes et propriétés qui va fournir une page Web côté client et qui sera donc la seule partie visible du site. Voici à quoi ressemble un script en React.js j'expliquerai ces différents points lorsque je parlerai de mon expérience.

import React, {Component} from 'react';

import {reduxForm} from'redux-form';

import SurveyForm from'./SurveyForm';

import SurveyFormReview from'./SurveyFormReview';

class SurveyNew extends Component{

state = {showReview: false};

renderContent(){

if(this.state.showReview){

return <SurveyFormReview

onCancel={() =>this.setState({showReview: false})}

/>;

}

return <SurveyForm

onSurveySubmit={() =>this.setState({showReview: true})}

/>;

}

render(){

return(

<div>

{this.renderContent()}

</div>

);

};

};

export default reduxForm({

form: 'surveyForm',

}) (SurveyNew);

* **Node.js** est un logiciel serveur qui va permettre d'utiliser JavaScript côté back-end c'est « la » structure de ces quatre fonctionnalités : sans lui les trois autres ne peuvent fonctionner vu qu'il est celui qui détient le langage nécessaire au fonctionnement de ceux-ci.

Durant mon stage j'ai suivi un cours qui montrait comment créer une application Web sous MERN sous la forme de plusieurs vidéos. Donc j'ai dû suivre le procédé comme il le fallait et assimiler toutes les composantes, fonctionnalités et rôles de chacun dans des cas différents.

Par exemple, MongoDB est intervenu pour enregistrer les données des personnes qui se connectaient à l'application via Google mais aussi pour l'utilisation de mongoose qui est une librairie pour MongoDB et qui permet de gérer les relations entre les données sous forme objet. Donc quand il fallait mettre en place des données on utilisait la librairie mongoose via MongoDB.

Express.js est intervenu lorsqu'il fallait utiliser et mettre le site sous le serveur local qui a généralement le port 3000. Grâce à lui on a pu voir un aperçu côté front-end du code écrit, il permet aussi d'utiliser des API via passport.js : c'est là qu'on a pris l'API pour la connexion à l'application via Google.

React.js est la bibliothèque qu'on a utilisé pour rendre le site visible sur le localhost 3000, grâce à ses méthodes qui sont aussi vastes qu’intéressantes. C'est la partie qui pour ma part m'a le plus plu et qui m'a donné envie d'en connaître davantage sur MERN.

C'est une bibliothèque assez complexe mais nécessitant peu de codes, notamment grâce aux méthodes qui lui appartiennent. Pour coder avec React.js il faut avoir une bonne base en JavaScript au préalable et se documenter sur tout ce qu'il propose.

En d’autres termes, il y a plusieurs composants à installer pour utiliser au mieux un site créé avec React.js et tout cela se fait via l'invite de commandes qu'il y a sous Windows. Pour manipuler les documents et fichiers ou exécuter des commandes, j'ai dû installer plusieurs bibliothèques qui sont compatibles avec React.js dans le but par exemple de faire apparaître un formulaire (la bibliothèque redux qui permet une interface pour l'utilisateur) ou du style CSS (le framework materialize-css qui va permettre d'apporter du style à notre site un peu comme bootstrap).

La syntaxe de React.js est assez compacte mais permet de faire des choses incroyables, comme dans le langage PHP où on peut importer des pages grâce à un « import » et travailler avec celles-ci mais pas que c'est justement comme ça qu'on importe les frameworks cités ci-dessus.

Ensuite on crée une classe qui aura les composants React et où on va insérer les méthodes propres à React.js. D'abord il y a la méthode render qui va retourner du JSX langage exclusif à React.js qui permet de mélanger du script HTML et du JavaScript et donc de montrer le contenu du site côté front-end et à l'intérieur on peut implémenter toutes les framework qu'on a importé au début du script et les utiliser comme bon nous semble.

Constructor est une méthode qui crée des objets avec une classe que l'on déclare avec super à l'intérieur de l'objet ; on peut déclarer une ou plusieurs variables pour nos prochaines méthodes qui auront certaines fonctions.

ComponentDidMount permet de monter tous les rendus fait, donc c'est la dernière pièce de notre code bien qu'on la place juste en dessous de constructor.

This.setState est une méthode qui met à jour les éléments de l'objet qu'on a placé dans constructor grâce à une fonction.

La méthode render permet de retourner en HTML (JSX) tout le composant c'est d'ailleurs la méthode par défaut et obligatoire qu'on retrouve dans un composant React.

Dès qu'on a fini on exporte le script en question dans le script qui est généré dès qu'on installe React.js sur notre ordinateur, qui se nomme App.js et qui est LA page qui sera mise en avant lors de l'apparition du site : c'est en somme la page principale vers laquelle on importera toutes les autres pages pour y créer une route vers celles-ci. Pour générer une page avec React.js il faut obligatoirement manipuler la console.

Enfin il y a Node.js qui a servi pour la base donc une fois Node.js installé tout le côté back-end est lié à l'utilisation de son langage de base : JavaScript.

* 1. Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité :
* 1.1. Maquetter une application

Tout d'abord avant de commencer je mets à disposition des définitions concernant certains termes techniques :

**Photoshop** : Logiciel de traitement et de retouche d'images et de photo produit par la société Adobe. [Photoshop](http://www.mosaiqueinformatique.com/formation-graphisme-infographie-pao-dao-nancy-lorraine/287-formation-photoshop-image-bitmap-infographie-pao-nancy-lorraine.html)est devenu le standard en matière de gestion des images matricielles (ou images "bitmap", constituées d'un "tapis de points"). Un logiciel tel qu'[Illustrator](http://www.mosaiqueinformatique.com/formation-graphisme-infographie-pao-dao-nancy-lorraine/formation-illustrator-debutant.html), lui, gère l'image numérique sous la forme de vecteurs (on parle alors d'images vectorielles).

**Wireframe** : Le wireframe, ou maquette fonctionnelle en français, est le schéma d’une page web ou d’une application. De par ses caractéristiques, c’est un outil qui apporte beaucoup d’avantages, surtout lors de la refonte ou de la création d’un site web.

**Nombre d'or** :  Nombre d'or, Section dorée, Divine proportion et autres appellations mystiques... sont des dénominations qui désignent un rapport arithmétique : le nombre d'or. Ce dernier n'est ni une mesure, ni une dimension, c'est un rapport entre deux grandeurs homogènes. Jean-Paul Delahaye affirme que le chemin des mathématiques à la numérologie est dangereux parce riche en interprétations...  En effet des milliers de pages ont été écrites sur le nombre d'or. Il serait connu depuis la nuit des temps. On le retrouve chez les peintres du début du siècle, dans les cathédrales gothiques, sur les façades des temples grecs et même au cœur de la Grande Pyramide. On dit qu'il aurait été transmis de bouche de pythagoricien à oreille d'initié, comme un secret universel et immuable (il n'était pas considéré comme un nombre puisque seuls les entiers sont des nombres chez les grecs). De nombreux tableaux seraient conçus selon les règles de la "divine proportion" (expression datant de 1509 avec Léonard de Vinci).  
Parmi les artistes de la Renaissance, Dürer est un de ceux qui connaissait les mathématiques. Il fit évoluer les proportions de "ses nus d'Adam et Eve", entre 1504 et 1507 après avoir été initié à la "secretissima scienta" par un maître dont il ne voulut pas révéler le nom mais qui fut sans doute le frère franciscain Luca Pacioli qui publia en 1494 la grande encyclopédie du XVe siècle.

**Les repères** : Les repères commentés vous permettent d’aligner des formes, des tranches et des sélections. Ils apparaissent automatiquement lorsque vous dessinez une forme, créez une sélection ou faites glisser. Vous pouvez masquer les repères commentés si nécessaire.

**La grille** : La grille est utile pour placer les éléments de manière symétrique. Elle s’affiche par défaut sous la forme de traits non imprimables, mais vous pouvez l’afficher sous la forme de points.

Les repères et la grille présentent des similitudes :

Les sélections, contours de sélection et outils sont attirés par le repère ou par la grille lorsqu’ils sont placés à moins de 8 pixels de trame (non d’image). Les repères sont également attirés par la grille lorsque vous les déplacez. Vous pouvez activer ou désactiver cette caractéristique de magnétisme.

L’espacement des repères, ainsi que la visibilité et le magnétisme des repères et de la grille sont propres à chaque image.

Le maillage de la grille, ainsi que la couleur et le style des repères et de la grille sont communs à toutes les images.

**Header** : le haut de la page l'emplacement du logo des liens vers les autres pages.

**Body** : le corps de la page c'est à dire son contenu.

**Footer** : le bas de page où on recense toutes les informations liées à l'entreprise (réseaux sociaux, pages annexes).

Pour la conception de la maquette j'ai décidé de faire un wireframe de 4 pages sur le logiciel Photoshop qui montrent les pages principales de mon site et d'apporter un premier aperçu visuel de celui-ci pour avoir une idée plus concrète de ce à quoi il devrait ressembler.

Les formes choisies sont principalement le rectangle en référence au nombre d'or pour avoir des proportions proches de la perfection. Pour m'aider je me suis servi des repères et de la grille pour pouvoir espacer au mieux les calques et les distinguer dans le champ de l'image.

Pour commencer j'ai créé un header, donc la barre de navigation, qui allait servir de modèle pour les 3 autres pages. J'y ai implémenté le logo, le titre du site, la « catchphrase », le bouton des pages et les boutons de connexion/déconnexion. Il est couvert d'un fond bleu cyan ce qui correspondra à la couleur de fond pour la majeure partie des pages, le fond en gris représente le fond lorsqu'il y a du texte par‑dessus, les textes en noir sont là comme dernier degré d'apparition dans cette nuance de couleur ce qui lui permet d'être l'élément le plus remarqué.

Ensuite j'ai créé un body qui lui est unique selon les pages, vu qu'il s'agit du contenu. Comme pour le header il a un fond de couleur bleu cyan, un fond pour les textes en gris et les textes de couleur noire. C'est un choix de couleurs voulu pour les mêmes raisons expliquées ci-dessus. La page d'accueil est dotée de plusieurs calques avec des rôles spécifiques bien défini qui sont répartis en trois zones bien distinctes, la page animaux a dans son body des éléments qui se ressemblent c'est pour montrer qu'il y aura ici une page qui affichera des éléments de même forme mais avec des informations différentes. Idem pour la page produits, quant à la page contacts elle a des calques qui forment tous ensemble un formulaire assez simple.

Enfin je crée le footer qui est bien différent des autres calques car il a un fond noir pour montrer la fin de la page et dire qu'il n’y aura plus rien après. On retrouve sinon les autres choix de couleurs qui sont le gris pour le fond de texte et le noir pour les textes, le blanc est utilisé à la toute fin pour être visible sur le fond noir tout simplement.

J'ai donc réparti ces différents calques en trois dossiers qui comportent les calques respectifs à chaque partie du wireframe de façon à bien les distinguer mais aussi à les réutiliser comme je l'ai fait pour le header et le footer.

* + 1.2. Réaliser une interface utilisateur web statique adaptable

**HTML** : Le HyperText Markup Language, généralement abrégé HTML ou dans sa dernière version HTML5, est le langage de balisage conçu pour représenter les [pages web](https://fr.wikipedia.org/wiki/Page_web). C’est un langage permettant d’écrire de l’[hypertexte](https://fr.wikipedia.org/wiki/Hypertexte), d’où son nom. HTML permet également de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d’inclure des ressources multimédias dont des [images](https://fr.wikipedia.org/wiki/Image_numérique), des formulaires de saisie et des programmes informatiques. Il permet de créer des documents [interopérables](https://fr.wikipedia.org/wiki/Interopérabilité_en_informatique) avec des équipements très variés de manière conforme aux exigences de l’[accessibilité du web](https://fr.wikipedia.org/wiki/Accessibilité_du_web). Il est souvent utilisé conjointement avec le [langage de programmation](https://fr.wikipedia.org/wiki/Langage_de_programmation) JavaScript et des feuilles de style en cascade (CSS). HTML est inspiré du Standard Generalized Markup Language (SGML). Il s'agit d'un format ouvert.

**CSS** : Le terme CSS est l'acronyme anglais de Cascading Style Sheets qui peut se traduire par "feuilles de style en cascade". Le CSS est un langage informatique utilisé sur l'internet pour mettre en forme les fichiers HTML ou XML. Ainsi, les feuilles de style, aussi appelé les fichiers CSS, comprennent du code qui permet de gérer le design d'une page en HTML.

**Bootstrap** : Bootstrap est un[framework](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203355-framework/) développé par l'équipe du réseau social Twitter. Proposé en open source (sous licence MIT), ce framework utilisant les langages [HTML](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203255-html-hypertext-markup-langage-definition-traduction/), CSS et JavaScript fournit aux développeurs des outils pour créer un site facilement. Ce framework est pensé pour développer des sites avec un design responsive, qui s'adapte à tout type d'écran, et en priorité pour les smartphones. Il fournit des outils avec des styles déjà en place pour des typographies, des boutons, des interfaces de navigation et bien d'autres encore. On appelle ce type de [framework](https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203355-framework/) un "Front-End Framework".

J'ai fait plusieurs scripts en HTML/CSS avec des finalités différentes et un contenu du même acabit. Pour commencer, j'ai écrit le script pour afficher la nav et le footer de ma page de base qui définit le HTML5 qui est « doctype ». A l'intérieur j’ai inséré plusieurs balises qui sont entre des chevrons < div>. La première balise est pour définir la langue à laquelle les personnes aveugles pourront écouter le site web.

La balise head est celle qui fournira les métadonnées qui incluront les caractères choisis dans meta charset= utf8, le titre dans la balise title, les indications dans la balise link qui comporteront les liens pour la bibliothèque bootstrap, le favicon et les scripts CSS pour les insérer dans notre script HTML

Ensuite nous avons la balise body où le contenu de la page Web sera visible grâce à mon script qui comportera plusieurs balises à commencer par la balise nav qui créera l’entête de la page Web suivi de la balise div qui va servir à créer un bloc où seront enfermé les éléments enfants.

Il y a la balise a qui va permettre de créer un lien qui mènera vers le chemin indiqué dans le href qui peut être un autre script ou bien une page web déjà existante en ligne comme google.com : dans mon cas il s'agit d'autres scripts que j'ai créé au préalable.

Puis nous avons la balise img qui ira chercher une image dans le dossier directement pour l'afficher dans la page Web. Pour la nav j'ai choisi de mettre le logo et pour le footer ce sont les logos des réseaux sociaux. On peut associer une balise a avec une balise img pour que l'image nous redirige vers la page inséré dans le href c'est une possibilité.

Pour fermer une balise, il faut tout simplement faire un chevron avec un slash après « nom de la balise » et remettre un chevron </div>. Une fois que je me suis occupé de la nav je suis passé au footer et le procédé est équivalent sauf qu'à la place de mettre nav on met footer pour définir que c'est le pied de page et à l'intérieur on y retrouve les mêmes balises que dans la nav. Une fois tout ceci fait nous aurons plusieurs blocs de textes qui se superposeront.

Voici le script que j'ai écrit pour faire apparaître la nav et le footer de mon site :

<!doctype html>

<html lang="fr">

<head>

<meta charset="utf-8">

<title>PETGAME</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

<link rel="shortcut icon" href="pics/monkey.png" type="image/x-icon">

<link rel="stylesheet" href="css/header.css">

<link rel="stylesheet" href="css/footer.css">

</head>

<body>

<nav>

<div class="head">

<div class="logo">

<a href="index.php"><img src="animals.png" style="width: 45px;"></a>

<p class="mt-2" style="font-size: 15px; font-weight: bolder; margin-bottom: 0px;">Petgame</p>

<p>Ici les animaux sont les maîtres !</p>

</div>

<div class="connexion">

<div class="lol">

<a href="login.php"><img src="pics/web.png" style="width: 15px; height: 15px; margin-right: 20px;">

<p style="margin-right: 20px;">Connexion</p></a>

</div>

<div class="lol">

<a href="logout.php"><img src="pics/web.png" style="width: 15px; height: 15px; ">

<p>Deconnexion</p></a>

</div>

</div>

</div>

<div class="menu" >

<a href="animaux.php">Animaux</a>

<a href="">Nos Produits</a>

<a href="http://animaux/wordpress/?page\_id=39">Contacts</a>

</div>

</nav>

<footer>

<div class="foot">

<div>

<h1 class="title">

Nos informations

</h1>

</div>

<div class="social">

<a href=""><img src="pics/tumblr-square.svg" style="width: 25px;" ></a>

<a href=""><img src="pics/twitter-square.svg" style="width: 25px;" ></a>

</div>

<div class="question">

<a href=""><p>Qui sommes-nous?</p></a>

<a href=""><p>Mentions légales</p></a>

<a href=""><p>Nos conseils</p></a>

</div>

</div>

<div class="foo">

<p>Credit Sidiki Camara Petgame &#xA9;</p>

</div>

</footer>

</body>

</html>

Ensuite j'ai ajouté du style grâce au langage CSS qui m'a permis de rendre mes deux gros blocs plus beaux et appréciables à la vue. J'ai voulu mettre en place un dégradé de couleur pour l'entête et le pied de page de mon site donc j'ai opté pour ça.

Puis je voulais 3 boutons simples donc je n’ai pas ajouté trop de styles voyants, je les ai juste bien positionnés grâce à la propriété display qui va permettre de mettre les éléments l'un à côté de l'autre et ensuite appliqué un justify-content qui va les espacer et obtenir la disposition souhaitée.

Pour rester dans la notion d'espacement nous avons deux propriétés basiques qui sont margin et padding. Le premier va espacer les éléments les uns des autres tandis que le second va espacer l'élément par rapport à son contenu. La syntaxe pour que le style ait une influence certaine sur l'élément voulu est de faire une appel à celui-ci dans un premier comme je l'ai fait pour la balise nav et de lui intégrer les propriétés à l'intérieur d'une accolade : balise{propriété CSS}.

Voici le script CSS de la nav :

nav{

background: linear-gradient(#FFFFFF, #9198e5);

border-bottom: 1px solid black;

}

.head{

display: flex;

justify-content: space-between;

margin-right: 15px;

margin-left: 15px;

}

.logo{

margin-top: 15px;

}

.connexion{

display: flex;

justify-content: space-between;

margin-top: 15px;

}

div.lol{

text-align: center;

}

.lol>a{

text-decoration: none;

color: black;

}

.menu{

margin: 0 auto;

text-align: center;

margin-bottom: 10px;

}

.menu>a{

text-decoration: none;

color: black;

margin: 15px;

padding: 5px;

background-color: grey;

}

Voici le script CSS du footer :

.foot{

background: linear-gradient(#FFFFFF, #9198e5);

border-bottom: 1px solid black;

}

.foo{

background: linear-gradient(#FFFFFF, #9198e5);

margin-bottom: 0px;

}

.foo>p{

text-align: center;

margin-bottom: 0px;

}

.title{

text-align: center;

/\* color: white; \*/

font-size: 25px;

}

div.social{

text-align: center;

margin-top: 15px;

}

div.social>a{

margin-left: 10px;

}

.question{

display: flex;

justify-content: space-around;

}

.question>a{

text-decoration: none;

color: black;

background-color: grey;

padding: 5px 5px 0px 5px;

margin-top: 15px;

margin-right: 35px;

margin-bottom: 10px;

}

Il y a aussi d'autres moyens d'implémenter du CSS je l'ai montré dans mon script HTML ou on le place directement dans la balise en écrivant « style » ou bien créer une balise <style></style> où on pourra écrire du CSS comme dans un script CSS de base mais il est plus conventionnel de les séparer même si je ne l'ai pas fait pour des raisons de précisions dans le visuel.

Voici le script de formulaire qui correspond à l'inscription des adhérents sur le site qui est dans une balise form avec deux instructions primordiales pour le côté back-end qui sont action et method.

Action est pour définir le script php qui se chargera de la bonne rentrée des données par l'utilisateur : le mien est register\_act.php et method est pour l'envoi des données au script mais qui ne seront pas visibles vue que l’instruction est post.

Ensuite dans ce formulaire j'ai mis différentes balises qui caractérisent un formulaire basique comme label, input et select. Le label est la légende, le titre qui sera associé au champ qui sera l’input de type (texte, date, email...) et le select est la liste déroulante qui proposera plusieurs choix à l'utilisateur.

Dans l'input on y trouve des attributs comme name ou pattern le premier servira de référence lorsque le formulaire sera envoyé côté back-end au niveau des données et le second quant à lui est uniquement utilisé côté front-end et sert à contraindre, selon les paramètres que j'ai mis, l'utilisateur à ne pas mettre des données que je juge erronées et hors limite : comme pour le nom où j'autorise uniquement des lettres et un tiret pour les prénoms composés avec une limite minimale de 3 caractères à 30 maximum.

<div class="container">

<h1 class="mt-3">Formulaire d'inscription</h1>

<form action="register\_act.php" method="post">

<div class="form-group">

<label for="fname">

Prenom

</label>

<input type="text" id="fname" name="fname" class="form-control" maxlength="30"

pattern="[A-Za-zàâäéèêëôöîïûü\- ]{3,30}" required>

</div>

<div class="form-group">

<label for="mail">Email</label>

<input type="email" name="mail" id="mail" required="" class="form-control">

</div>

<div class="form-group">

<label for="mail">Numéro de téléphone</label>

<input type="text" name="phone" id="phone" class="form-control">

</div>

<div class="form-group">

<label for="dob">Date de naissance</label>

<input type="date" name="dob" id="dob" pattern="[0-9]{4}-(0[1-9]|1[012])-(0[1-9]|1[0-9]|2[0-9]|3[01])" class="form-control">

</div>

<div class="form-group">

<label for="reg\_id" class="mt-3">Région: </label>

<select id="reg\_id" name="reg\_id" class="form-control">

<option>--Sélectionnez votre région--</option>

<?php

$val = $conn->query('SELECT reg\_id, regionName FROM region ORDER BY reg\_id');

$html='';

foreach ($val as $row1) {

$html .= '<option value="'.$row1['reg\_id'].'">'.$row1['regionName'].'</option>';

}

echo $html;

?>

</select>

</div>

<div class="form-group">

<label for="type">Particulier ou Propriétaire:</label>

<select name="type" id="type" class="form-control">

<option value="type">

---Faites votre choix---

</option>

<option value="0">Particulier</option>

<option value="1">Propriétaire</option>

</select>

</div>

<div class="form-group">

<label for="gender">Genre</label>

<select name="gender" id="gender" class="form-control">

<option value="">

---Faites votre choix---

</option>

<option value="F">Féminin</option>

<option value="M">Masculin</option>

<option value="N">Neutre</option>

</select>

</div>

<input type="submit" value="S'inscrire" class="btn btn-info mb-3">

</form>

</div>

* 1.3. Développer une interface utilisateur web dynamique

**JavaScript** : JavaScript est un langage de programmation qui permet d’implémenter des mécanismes complexes sur une page web. À chaque fois qu’une page web fait plus que simplement afficher du contenu statique afficher du contenu mis à jour à des temps déterminés, des cartes interactives, des animations 2D/3D, des menus vidéo défilants, etc... JavaScript a de bonnes chances d’être impliqué. C’est la troisième couche des technologies standards du web, les deux premières (HTML et CSS).

J'ai intégré du JavaScript dans mon projet en utilisant une requête AJAX dans un premier temps qui appelle un script PHP contenant une requête qui va chercher l'id region se trouvant dans le script HTML sous la forme d’une barre de recherche puis répondre à un événement blur qui va aller chercher dans une requête SQL la région relier à celle que l'utilisateur aura écrite dans la barre de recherche pour ensuite filtrer les éléments et faire apparaître dans la page web uniquement les éléments reliés à la région écrite.

Ensuite nous avons un second script qui renvoie un écouteur d'événement qui pendant le chargement de la page va exécuter un événement une fois qu'on cliquera sur un bouton dans mon cas il s'agit du bouton supprimer qui au déclenchement du bouton enverra un message pour confirmer si l'utilisateur veut valider la suppression de l'annonce.

document.getElementById('region').addEventListener('blur', function() {

// Requête AJAX pour lecture dans BDD

let xhr = new XMLHttpRequest();

xhr.open('get', 'animaux\_ajax.php?region=' +this.value, true);

xhr.addEventListener('readystatechange', function() {

if (xhr.status === 200 && xhr.readyState === 4) {

document.getElementById('animaux').innerHTML = xhr.responseText;

}

}, false);

xhr.send();

});

window.addEventListener(

'load',

function(){

let buttons = document.querySelectorAll('a.btn-danger');

for(let i=0; i < buttons.length; i++){

buttons[i].addEventListener(

'click',

function(evt){

evt.preventDefault();

let answer = confirm('Voulez-vous vraiment supprimer cette ligne ?');

if(answer){

location.href = evt.target.href;

}

},

false

);

}

},

false

);

* 1.4. Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Définitions :

**Wordpress** : WordPress est un système de gestion de contenu (SGC ou content management system (CMS) en anglais) gratuit, libre et open-source. Ce logiciel écrit en PHP repose sur une base de données MySQL et est distribué par l'entreprise américaine Automattic. Les fonctionnalités de WordPress lui permettent de créer et gérer différents types de sites Web : site vitrine, site de vente en ligne, site applicatif, blogue, ou encore portfolio. Il est distribué selon les termes de la licence GNU GPL version 2. Le logiciel est aussi utilisé comme socle du service multisite WordPress.com, celui-ci supporte plusieurs millions de sites.

J'ai fait une partie WordPress dans mon projet qui s'apparente aux pages « produits » et « contacts ». Dans cette deuxième partie du site tout a été fait à partir de Wordpress et pour cela j'ai installé des fonctionnalités qui sont compatibles avec le CMS WordPress : il y a les thèmes qui sont propres à WordPress qu'on peut installer directement ou bien créer sois‑même - on appellera donc cela un thème enfant qui restera tel quel, peu importe le nombre de mises à jour qu'il y aura sur WordPress étant donné qu'il sera écrit sur le serveur local.

Pour créer un thème il faut mettre en commentaire dans un script CSS les paramètres dont on aura besoin : son nom, le texte du domaine lors de son apparition sur WordPress, le template donc le thème parent qui est twentytwenty dans mon script, sa version qui peut être configurée comme on le veut, une description, des tags pour le trouver plus facilement lors des recherches, son auteur et les licences GNU et URI.

Voici le script du thème que j'ai créé à partir du template de twentytwenty:

/\*

Theme Name: Twenty Twenty Child

Text Domain: twentytwentychild

Template: twentytwenty

Version: 1.0

Requires at least: 4.7

Requires PHP: 5.2.4

Description: twenty twenty child

Tags: blog, one-column, custom-background, custom-colors, custom-logo, custom-menu, editor-style, featured-images, footer-widgets, full-width-template, rtl-language-support, sticky-post, theme-options, threaded-comments, translation-ready, block-styles, wide-blocks, accessibility-ready

Author: Sidiki

Author URI: http://baobab-ingenerie.fr

Theme URI: http://baobab-ingenerie.fr

License: GNU General Public License v2 or later

License URI: http://www.gnu.org/licenses/gpl-2.0.html

All files, unless otherwise stated, are released under the GNU General Public

License version 2.0 (http://www.gnu.org/licenses/gpl-2.0.html)

This theme, like WordPress, is licensed under the GPL.

Use it to make something cool, have fun, and share what you've learned

with others.

\*/

Puis il y a la fonctionnalité qu'on appelle plugin, via ces plugins j'ai pu définir mes deux pages.

La page « produits » est gérée par le plugin nommé WooCommerce qui va créer toute la partie e-commerce avec comme but de montrer les produits animaliers (alimentation, jouets, objets...) et l'utilisateur pourra acheter des produits directement sur mon site via le plugin WooCommerce.

Comme pour les thèmes, on peut aussi créer son propre plugin sauf que le procédé est un peu plus complexe même si le début est identique dans mon projet. J'ai créé un plugin pour faire un formulaire de contact qui permettra à l'utilisateur d'envoyer un message à l'administrateur donc moi sur n'importe quel sujet lié au site (erreur de produits, problèmes dans une transaction avec un autre particulier ou autres...).

Dans un script PHP cette fois-ci on mettra en commentaire les paramètres puis pour le coup il faudra rajouter des éléments en plus et faire une fonction qui va permettre d'activer et désactiver le plugin dans les extensions. Une autre fonction va mettre en place le formulaire et dans ce formulaire on va refaire une fonction pour assainir le champ de texte lorsque l'utilisateur aura écrit dessus et enfin un shortcode devra être inséré pour utiliser pleinement ce plugin grâce à la méthode add\_shortcode.

/\*

Plugin Name: Sidplug formulaire de contact

Plugin URI:

Description: un exemple de formulaire de contact

Version: 1.0

Author: Sidiki

Author URI:

\*/

function Sidplug\_form\_contact\_activate() {

Sidplug\_create\_contact\_table();

}

register\_activation\_hook( \_\_FILE\_\_, 'Sidplug\_form\_contact\_activate' );

function baobab\_form\_contact\_deactivate() {

}

register\_deactivation\_hook( \_\_FILE\_\_, 'Sidplug\_form\_contact\_deactivate' );

function Sidplug\_create\_contact(){

// sanitize form values

$contact = array();

$contact['nom'] = sanitize\_text\_field( $\_POST["cf-name"] );

$contact['email'] = sanitize\_email( $\_POST["cf-email"] );

$contact['sujet'] = sanitize\_text\_field( $\_POST["cf-subject"] );

$contact['message'] = esc\_textarea( $\_POST["cf-message"] );

return $contact;

}

add\_shortcode( 'Sidplug\_contact\_form', 'cf\_shortcode' );

* 2.1. Créer une base de données

Définitions :

**Base de données** : Une base de données, permet de stocker et de retrouver l'intégralité de données brutes ou d'informations en rapport avec un thème ou une activité ; celles-ci peuvent être de natures différentes et plus ou moins reliées entre elles.

**Base de donnée relationnelles** : Une base de données relationnelle est un type spécifique de base de données. Ce qui distingue une base de données relationnelle de tout autre type de base de données, c’est qu’elle permet de mettre en relation différentes données au sein de cette dernière.

**Dictionnaire de données** : Un dictionnaire des données est une collection de métadonnées ou de données de référence nécessaire à la conception d'une base de données relationnelle. Il revêt une importance stratégique particulière, car il est le vocabulaire commun de l'organisation.

**Modèle conceptuel de données** : Le modèle conceptuel des données (MCD) a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information. Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités.

Pour la création de la base de données j'ai d'abord établi un dictionnaire de données qui recense toutes les entités dont je vais avoir besoin dans mon projet et avoir une idée de que je pourrai y intégrer comme valeurs.

Donc le nom de ma base de données est comme celle de mon site « petgame » et dedans j'ai inséré quatre tables qui sont « region » pour y insérer les régions de France, « pet » pour ajouter les animaux domestiques, « contacts » pour l'inscription des adhérents qui peuvent être particulier ou propriétaire, « generique » pour y mettre les différents types d'animaux (chat, chient, lapin, perroquet, hamster...).

Avec ces quatre tables j'ai pu gérer tout le côté back-end de mon site notamment grâce aux colonnes qui ont grandement joué dans mes scripts.

* Pour la table « region » il y a :
  + reg\_id qui est l'id de type integer qui est une clé primaire, non vide et auto incrémentale ;
  + regionName qui est le nom de la région de type varchar non vide aussi.
* Pour la table « contacts » il y a :
  + cli\_id qui est l'id de type integer, une clé primaire, non vide et auto incrémental ;
  + fname qui est le nom de l'adhérent de type varchar non vide ;
  + gender qui est le genre de l'adhérent de type enum ;
  + dob qui est la date de naissance de l'adhérent de type date et non vide ;
  + type qui est de si l'adhérent est un particulier ou un propriétaire de type enum et non vide ;
  + phone qui correspond au numéro de téléphone de l'adhérent de type varchar ;
  + mail qui est l'adresse email de l'adhérent de type varchar et non vide ;
  + pass qui est le mot de passe de l'adhérent de type varchar et non vide.
* Pour la table « pet » il y a :
  + pet\_id qui est l'id de type integer, une clé primaire, non vide et auto incrémental ;
  + fname qui est le nom de l'animal de type varchar, non vide ;
  + race qui est la race de l'animal (s'il en a une) de type varchar ;
  + gender qui est le sexe de l'animal qui est de type enum ;
  + photo qui est la photo de l'animal de type longblob ;
  + description qui est la description que le propriétaire fera de l'animal de type longtext.
* Pour la table « generique » il y a :
  + gen\_id qui est l'id de type integer, une clé primaire, non vide et auto incrémental ;
  + nom qui est le nom du type de l'animal de type char.

Toutes ces tables avec leurs colonnes respectives auront un rôle bien défini et pourront pleinement être utiles dans chaque partie de mon projet. En amont j'ai défini un Modèle Conceptuel de Données (MCD) qui va me permettre de mettre en relation mes tables de sorte à créer une affinité entre celles-ci qu'il y ait une harmonie optimale et qu'elles puissent fonctionner au mieux.

* La table « contacts » aura une relation de 1 à N avec la table « region » étant donné qu'une région peut contenir 1 à plusieurs adhérents tandis qu'un adhérent ne peut être contenu que dans une seule région.
* La table « pet » aura trois relations
  + une avec la table « contacts » qui sera une relation de 0 à N vu qu'un animal domestique appartient à un propriétaire et qu'un adhérent peut avoir entre zéro et plusieurs animaux vu qu'il peut s'agir d'un particulier qui veut sans doute obtenir son premier animal de compagnie.
  + Il y a aussi une relation de 1 à N avec la table « generique » car un nom générique peut être porté que par un ou plusieurs animaux mais un animal ne peut qu'avoir un seul nom générique.
* 2.3. Développer la partie back-end d’une application web ou web mobile

Définitions :

**Wampserver** : WampServer est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant trois serveurs, un interpréteur de script, ainsi que phpMyAdmin pour l'administration Web des bases MySQL.

**MySQL Workbench** : MySQL Workbench (anciennement MySQL administrator) est un logiciel de gestion et d'administration de bases de données MySQL créé en 2004. Via une interface graphique intuitive, il permet, entre autres, de créer, modifier ou supprimer des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour ce faire, il doit être connecté à un serveur MySQL.

Pour créer une base de données sur le serveur local j'ai utilisé l'environnement Wampserver et le logiciel MySQL Workbench. En installant Wampserver j'ai pu lancer un serveur local sur lequel j’ai installé des logiciels qui pourront créer, lire, mettre à jour et supprimer mes données.

Le logiciel en question est MySQL Workbench qui permet de créer un Modèle Physique de Données (MPD) déduit du Modèle Conceptuel de Données et à partir duquel on mettra en place notre base de données et ses tables.

Dans mon projet comme expliqué ci-dessus je créerai ma base de données « petgame » avec les quatre tables « region », « contacts », « pet » et « generique » et avec Workbench je ferai un diagramme qu'on appelle Modèle Physique de Données pour enfin définir les relations entre les tables.

Le changement réside dans les relations où nous n’auront plus des verbes qui montreront ces liens mais la clé primaire de la table qui aura la cardinalité la moins forte recevra la clé primaire de la table qui a la cardinalité la plus forte dans sa table et deviendra une clé étrangère. Donc si je récapitule avec plus de détails :

* Dans la table « contacts » on aura la clé primaire de table « region » vu que la cardinalité de la table « region » est plus forte que celle de la table contacts avec une relation de 1 à N.
* Dans la table « pet » on aura les clés primaires des tables « contacts » et « generique » pour la première car « contacts » a une cardinalité plus forte de 0 à N et generique de 1 à N.

Quand cela a été fait, j'ai procédé à la transformation du Modèle Physique de Données en script SQL pour les enregistrer dans la base de données. Pour ce faire j'ai fait un forward engineer et ça m'a retourné un script SQL prêt à l’emploi avec les relations établies au préalable dans le diagramme de données.

Voici le script SQL :

CREATE SCHEMA IF NOT EXISTS `petgame` DEFAULT CHARACTER SET utf8 COLLATE utf8\_bin ;  
USE `petgame` ;  
  
 Table `petgame`.`region`  
  
CREATE TABLE IF NOT EXISTS `petgame`.`region` (  
 `reg\_id` INT NOT NULL AUTO\_INCREMENT,  
 `regionName` VARCHAR(50) NOT NULL,  
 PRIMARY KEY (`reg\_id`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8  
COLLATE = utf8\_general\_ci  
;

Table `petgame`.`contacts`  
  
CREATE TABLE IF NOT EXISTS `petgame`.`contacts` (  
 `cli\_id` INT NOT NULL AUTO\_INCREMENT,  
 `fname` VARCHAR(30) NOT NULL,  
 `gender` ENUM('F', 'M', 'N') NULL DEFAULT NULL,  
 `dob` DATE NULL DEFAULT NULL,  
 `type` ENUM('0', '1') NOT NULL,  
 `mail` VARCHAR(100) NOT NULL,  
 `pass` VARCHAR(50) NOT NULL,  
 `reg\_id` INT NOT NULL,  
 PRIMARY KEY (`cli\_id`),  
 UNIQUE INDEX `mail` (`mail` ASC),  
 INDEX `fk\_contacts\_region1\_idx` (`reg\_id` ASC),  
 CONSTRAINT `fk\_contacts\_region1`  
 FOREIGN KEY (`reg\_id`)  
 REFERENCES `petgame`.`region` (`reg\_id`)  
 ON DELETE NO ACTION  
 ON UPDATE NO ACTION)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8  
COLLATE = utf8\_general\_ci  
;  
  
INSERT INTO `contacts` (`cli\_id`, `fname`, `gender`, `dob`, `type`, `mail`, `pass`, `reg\_id` ) VALUES  
(1, 'FJack', 'M', '1997-04-17', '1', 'test@outlook.fr', 'Toto', 3);

Table `petgame`.`generique`  
  
CREATE TABLE IF NOT EXISTS `petgame`.`generique` (  
 `gen\_id` INT NOT NULL AUTO\_INCREMENT,  
 `nom` CHAR(30) NULL DEFAULT NULL,  
 PRIMARY KEY (`gen\_id`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8  
COLLATE = utf8\_general\_ci  
;  
  
INSERT INTO generique (nom) VALUES ('Chien');  
INSERT INTO generique (nom) VALUES ('Chat');  
INSERT INTO generique (nom) VALUES ('Perroquet');  
INSERT INTO generique (nom) VALUES ('Souris');  
INSERT INTO generique (nom) VALUES ('Lapin');  
INSERT INTO generique (nom) VALUES ('Hamster');  
INSERT INTO generique (nom) VALUES ('Ecureuil');  
  
Table `petgame`.`pet`

CREATE TABLE IF NOT EXISTS `petgame`.`pet` (  
 `pet\_id` INT NOT NULL AUTO\_INCREMENT,  
 `fname` VARCHAR(30) NOT NULL,  
 `race` VARCHAR(100) NULL DEFAULT NULL,  
 `gender` ENUM('F', 'M') NULL DEFAULT NULL,  
 `photo` LONGBLOB NULL DEFAULT NULL,  
 `description` LONGTEXT NULL DEFAULT NULL,  
 `gen\_id` INT NOT NULL,  
 `cli\_id` INT NOT NULL,  
 `reg\_id` INT NOT NULL,  
 PRIMARY KEY (`pet\_id`),  
 INDEX `fk\_pet\_generique\_idx` (`gen\_id` ASC),  
 INDEX `fk\_pet\_contacts1\_idx` (`cli\_id` ASC),  
 INDEX `fk\_pet\_region1\_idx` (`reg\_id` ASC),  
 CONSTRAINT `fk\_pet\_contacts1`  
 FOREIGN KEY (`cli\_id`)  
 REFERENCES `petgame`.`contacts` (`cli\_id`)  
 ON DELETE NO ACTION  
 ON UPDATE NO ACTION,  
 CONSTRAINT `fk\_pet\_generique`  
 FOREIGN KEY (`gen\_id`)  
 REFERENCES `petgame`.`generique` (`gen\_id`)  
 ON DELETE NO ACTION  
 ON UPDATE NO ACTION,  
 CONSTRAINT `fk\_pet\_region1`  
 FOREIGN KEY (`reg\_id`)  
 REFERENCES `petgame`.`region` (`reg\_id`)  
 ON DELETE NO ACTION  
 ON UPDATE NO ACTION)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8  
COLLATE = utf8\_general\_ci  
;  
INSERT INTO `pet` (`pet\_id`, `fname`, `gen\_id`, `cli\_id`, `race`, `gender`, `photo`,`description`, `reg\_id`) VALUES  
(1, 'Felix', '5', '1', 'Labrador', 'M', NULL, NULL, 4);

* 2.3. Développer les composants d’accès aux données

**PHP** : PHP: Hypertext Preprocessor, plus connu sous son sigle PHP (sigle auto-référentiel), est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.

Après avoir mis en place ma base de données sur le serveur j'ai pu l'utiliser dans mes scripts PHP. Pour commencer je me suis connecté à celle-ci en utilisant l'objet new PDO qui permet de se connecter à une base de données donc j'ai pu rentrer les coordonnées de l'hôte, du nom de la BDD, le caractère utilisé et l'identifiant et mot de passe de mon environnement après ça je pouvais faire appel à toutes les tables de la BDD petgame comme bon me semblait.

<?php

try{

$conn = new PDO(

'mysql:host=localhost;dbname=petgame;charset=utf8',

'root',

''

);

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$conn->setAttribute(PDO::ATTR\_DEFAULT\_FETCH\_MODE, PDO::FETCH\_ASSOC);

}catch(PDOException $err){

echo '<p class="alert alert-danger">'.$err->getMessage();

}

?>

Une fois connecté à la base de données, j'ai créé un autre script qui va gérer côté back-end l'inscription des adhérents si les valeurs ont bien été insérées par celui-ci et pour se faire j'ai importé la connexion à ma BDD dans mon script grâce à un include\_once accompagné du nom du script et une fois cela fait j'ai pu faire une requête SQL qui fera appel à la table contacts et regardera la colonne mail.

Puis je mets en place des paramètres qui ramèneront le mail inséré par l'utilisateur et le mettre dans la base de données. Donc je prépare la requête avec la méthode prepare qui protège contre les injections de code SQL, ensuite j’exécute la requête pour ramener les données.

Avant l'insertion des données j’émets une condition qui dit que si dans la base de données il y a un doublon concernant la colonne mail que l'inscription ne se fasse pas autrement l'inscription sera autorisée et une fois les valeurs entrées on les protègent avec la méthode htmlspecialchars qui évite les injections HTML et Javascript.

Une fois tout ça fait sans problème l'utilisateur recevra un mail qui le rendra adhérent au site.

<?php

include\_once 'db\_connect.inc.php';

include\_once 'fonctions.inc.php';

$sql = 'SELECT COUNT(\*) AS Nb FROM contacts WHERE mail = ?';

$params = array($\_POST['mail']);

$data = $conn->prepare($sql);

$data->execute($params);

$row = $data->fetch();

if((int) $row['Nb'] === 0){

$sql = 'INSERT INTO contacts(fname, mail, phone dob, gender, type, pass, reg\_id) VALUES(:fname, :mail, :dob, :gender, :type, :phone, :pass, :reg\_id)';

$data =$conn->prepare($sql);

$pass = get\_password(); //'basique';

$hash = sha1(md5($pass).sha1($\_POST['mail']));

$params = array(

':fname' => htmlspecialchars($\_POST['fname']),

':mail' => htmlspecialchars($\_POST['mail']),

':dob' => htmlspecialchars($\_POST['dob']),

':gender' => htmlspecialchars($\_POST['gender']),

':type' => htmlspecialchars($\_POST['type']),

':phone' => htmlspecialchars($\_POST['phone']),

':reg\_id' => htmlspecialchars($\_POST['reg\_id']),

':pass' => $hash

);

$data->execute($params);

$html = '<p>Bienvenue ' .$\_POST['fname'].',';

$html .= '<p>Nous vous confirmons votre inscription à Petgame, vous pouvez désormais vous connecter en tant que membre en utilisant les accréditations suivantes :';

$html .= '<ul>';

$html .= '<li> Identifiant:' .$\_POST['mail'];

$html .= '<li> Mot de passe:' .$pass;

$html .= '</ul>';

$html .= '<p> A très vite sur PETGAME';

$header = "MIME-Version: 1.0 \n"; // Version MIME

$header .= "Content-type: text/html, charset=utf8 \n"; // Format du mail

$header .= "From: sidcoma@outlook.fr \n"; //Expéditeur

$header .= "Reply to: grec@faim.com \n";

$header .= "Disposition-Notification-To: cocorico@poule.fr \n"; // Accusé de reception

$header .= "X-Priority: 1\n";

$header .= "X-MSMailPriority: High \n";

mail($\_POST['mail'], 'Confirmation inscription', $html, $header);

header('location: index.php');

echo '<p>OK';

}else{

echo 'L\'adresse mail existe déjà dans la base de données !';

}

?>

Ici j'ai mis un bout de script qui montre l'aspect update géré dans mon site. En effet l'adhérent, s'il s'agit du propriétaire, pourra accéder à la liste des annonces et dans celle-ci il pourra modifier l'annonce qu'il a mise en ligne et grâce à la requête ci-dessous la modification sera gérée.

<?php

$sql = 'UPDATE pet SET fname=:fname, gender=:gender, description=:description, gen\_id=:gen\_id, reg\_id=:reg\_id, cli\_id=:cli\_id, race=:race, photo=:photo WHERE pet\_id ='.$\_GET['pet\_id'];

$data = $conn->prepare($sql);

$data->execute($params);

} catch (PDOException $err) {

echo $err->getMessage();

}

header('location:animaux.php');

?>

Enfin voici le script pour la suppression d'annonces que le propriétaire pourra aussi faire dans la liste qui sera gérée par la requête delete toujours avec le même principe que les autres scripts SQL et une fois cela fait l'annonce disparaîtra côté front-end et back-end.

<?php

if(isset($\_GET['pet\_id']) && !empty($\_GET['pet\_id'])){

try{

include\_once 'db\_connect.inc.php';

$sql= 'DELETE FROM pet WHERE pet\_id = ?';

$params = array($\_GET['pet\_id']);

$data = $conn->prepare($sql);

$data->execute($params);

unset($conn);

header('location: annonce\_list.php');

} catch(PDOException $err){

echo '<p>'.$err->getMessage().'</p>';

}

}

?>

Voilà la fin de mon mémoire de projet en vous remerciant de votre lecture et que vous l'avez apprécié.

**Sidiki Camara**